Building Memristor Applications: From Device Model to Circuit Design

Fernando García-Redondo, Marisa López-Vallejo, and Pablo Ituero

***Abstract*—Since the memristor was first built in 2008 at HP Labs, no end of devices and models have been presented. Also, new applications appear frequently. However, the integration of the device at the circuit level is not straightforward, because available models are still immature and/or suppose high computational loads, making their simulation long and cumbersome. This study assists circuit/systems designers in the integration of memristors in their applications, while aiding model developers in the validation of their proposals. We introduce the use of a memristor application framework to support the work of both the model developer and the circuit designer. First, the framework includes a library with the best-known memristor models, being easily extensible with upcoming models. Systematic modifications have been applied to these models to provide better convergence and significant simulations speedups. Second, a quick device simulator allows the study of the response of the models under different scenarios, helping the designer with the stimuli and operation time selection. Third, fine tuning of the device including parameters variations and threshold determination is also supported. Finally, SPICE/Spectre subcircuit generation is provided to ease the integration of the devices in application circuits. The framework provides the designer with total control overconvergence, computational load, and the evolution of system variables, overcoming usual problems in the integration of memristive devices.**

***Index Terms*—Design framework, memristor, process variations, simulation, spice.**

I. Introduction

POSTULATED by Chua in 1971 [1], a memristor is the fourth basic electrical component—along with resistors, capacitances, and inductances—displaying a predictable relationship between its resistance and the electrical charge traversing it. Depending on the direction of the current flowing through the device, its resistance increases or decreases, and when stopped, the memristor stores the final resistance value. It was not until 2008 that the first actual implementation of a memristor saw light, developed by HP Labs employing TiO_2 [2]. Other groups have since been working on the development of new memristive structures [3]–[5], and the industry and the scientific community are endeavoring to find novel applications for these devices and achieve precise models that describe their behavior [6]–[13].

One of the main engines of the quick development of novel memristive devices has been its application to electronic systems with a great potential to overcome existing solutions. In the field of random access memories (RAMs), memristors have been used to store a resistance value in each cell, what is known as a resistive RAM (RRAM) [14], [15].

These memories, employing the appropriate read and write drivers, can store multilevel values and can achieve higher densities and speeds than static CMOS RAMs; furthermore, they do not require any energy to maintain the stored data. Memristors, in contrast to the other passive components or transistors, have been successfully employed to simulate the behavior of neurons, making them very attractive in the area of neural networks [16]. Also, concerning field programmable gate arrays, the memory characteristics of the memristors have been employed to build effective lookup tables and their varying resistiveness has led to the emergence of new switching matrix structures [17]. In the analog field, designers are using memristors to build circuits with controllable thresholds [18].

Modeling, simulation, and prediction of memristor's behavior are fundamental pieces of the development and validation of new applications. In 2008, after the first device was implemented, HP Labs introduced the first model based upon the few available fitting points from their memristor [2]. In the following years, other works put forward improved models that better tracked the dynamic characteristics of that device. In 2011, HP Labs presented the first model that departed from a physical formulation—the tunnel effect—significantly improving the accuracy [10]; at the same time, new devices began to appear and their models also employed physics-based formulations. In the last few years, several patents have presented new memristive structures, and the modeling community is working toward generalized models that are able to predict any device.

In spite of the big effort carried out by model developers, circuit and systems designers still have serious difficulties to integrate and validate memristors. Models are immature and controlling convergence, accuracy, speed, and computational load of simulations is especially hard and tricky. As it is normal at the early stages of a technology, especially when dealing with nanostructures, there is still a gap between technologists, model developers, and circuit designers. In an attempt to bridge this gap, Biolek *et al.* [13] presented in 2013, several models of memory devices described in Simulation program with integrated circuit emphasis (SPICE), the traditional standard for circuit designers; they provided with interesting features such as the possibility to control the time step, soft transition functions,

simulation parameters control, and behavioral integrators. Still, the approach is not generalizable, it just includes three models and the designer needs to go through several trial-and-error phases to select the correct time step and control integral limits.

The present study advances in that direction and aspires to help circuit/system designers in the integration of the memristor, while at the same time aiding model developers in the validation of their proposals. We introduce the memristor application framework (MAF), hereafter that encompasses three modules corresponding to the stages in the standard work methodology: device model characterization and validation for a given application scenario; memristor level characterization, computation and selection of stimuli and operation time; SPICE/Spectre subcircuit generation. The main features of the MAF that contribute to the state of the art are as follows.

1) It includes the seven best-known memristor models and future upcoming models will be easy to introduce.
2) It provides both SPICE and Spectre—the de facto industrial standard from Cadence—subcircuits.
3) It employs systematic modifications to improve SPICE code—i.e., aggregation/disaggregation of sources, aggregation of variables, etc.—that are translated into significant simulations speedups.
4) The MAF allows the analysis of parameter variations, an important aspect in memristor validation. The memristor, as well as other nanodevices, is highly affected by variability due to its extremely nonlinear behavior.
5) The framework provides the designer with total control over convergence, computational load, and the evolution of system variables.

The MAF is available for download at *www.vlsi.die.upm.es/memristor*.

The structure of the rest of this paper is as follows. First, we describe the library of memristor models included. The next section collects the proposed systematic improvements related to the subcircuits netlist. Section IV describes the framework, focusing on the main modules and functionalities, but also proposing a design methodology and presenting an example of use. In Section V, we detail the performance improvements achieved by the subcircuit modifications. Some concluding remarks can be found in Section VI.

II. Review of the Memristor Models

Let us start with a review of the seven most important memristor models that have been presented in the literature to date, ordered by date of publication. A memristor model includes a description of the device behavior, usually in the form of differential equations, defining the specific method to compute the response to a given stimulus. Additionally, the model also includes device parameters, a set of magnitudes and state variables, and simulation control variables. Table I details the characteristics of each model. Next, we briefly describe their highlights.

Linear ion drift [2] is a generic charge-controlled memristor. This model, whose state variable linearly depends on the current, can be expressed through a polynomial equation system. Additionally, this study introduces the nonlinear drift model, which is described more deeply in later works.

Nonlinear ion drift models using different window functions [2], [7], [9], [19]. Several memristor-based applications are developed using these models [14], [15], which include nonlinearity by shaping state variable using window functions. Some of these functions have limitations, like the *hard switching condition*, when modeling the boundary conditions [2], [23], or lack of modeling the threshold effect in the memristor behavior.

Simmons tunneling barrier model [10], [20] is the most complete memristor model, built as a metal/insulator/metal (MIM) junction, combined with an in-series resistance. The structure behaves following the Simmons tunneling theory [21], [22], matching the experimental results [10]. This model often experiences convergence problems.

Yakopcic's model [8] is characterized by a state variable exponential drift and hyperbolic sinusoid shape, following and simplifying the previous MIM memristors. This model, which fits the characteristic $i(v)$ curves of several physical devices [24]–[27], is proved to be useful when developing the memristor-based neuromorphic systems.

TEAM model [12]. The Threshold Adaptive Memristor model also derives from the Simmons tunneling barrier model. Like Yakopcic's model, the TEAM model proposes a simplification of the complex MIM structure presented in [10] and [20]. In contrast to Yakopcic's hyperbolic sine *i–v* relation, this TEAM model is based on a polynomial curve, and consequently, reduces the computational load.

Simplified Simmons barrier model, presented in [12], is a modification of the Simmons tunneling barrier. The *i–v* relation is simplified, reducing the complex system presented in [20] to a single exponential relation. Therefore, the performance is improved and most convergence problems are solved, while on the other hand, the accuracy is reduced.

Eshraghian MIM and threshold model [11]. The latest model that appeared in the literature presents a complete and deep study of the underlying mechanisms of the memristor dynamics. Based on this study, Eshraghian *et al.* present an accurate, highly customizable, and robust (from the convergence and overflow point of view) memristor model.

Circuit designers who wish to introduce one of these models in their work flow must go through a series of stages before they can actually cosimulate and validate the memristors along with other circuital elements. Unless provided by the model developer, they must write a SPICE netlist based on the model equations. Then, there are several variables that need careful adjustment—due to the nonlinearity nature inherent to the models—to match a specific device. Also, the range and type of the stimuli feeding the memristor have a big impact on its behavior and require thorough analysis. Convergence, precision, and computational load are three key factors for any circuital simulation that in the case of these immature models are especially hard to control. Next sections describe our proposal to help circuit designers in these tasks.

III. Systematic Improvement of Model Netlists

The authors of some of the models described in Section II did not provide with a direct circuital realization. We have developed netlists describing all the models not only in SPICE,

TABLE I
REVIEW OF MEMRISTOR MODELS PRESENTED IN THE LITERATURE

Model	State variable $(x(t), w(t))$ evolution	$i(t) - v(t)$ relationship
Linear ion drift [2]	$\frac{dw}{dt} = \mu_v \frac{R_{\rm on}}{D} i(t)$	$v(t) = \left\{ R_{\rm on} * \frac{w(t)}{D} + R_{\rm off} * \left(1 - \frac{w(t)}{D}\right) \right\} i(t)$
Non-linear ion drift [2], [7], [9], [19]	$\frac{dw}{dt} = \mu_v \frac{R_{\rm on}}{D} i(t) f(w)$, where $f(w) = 1 - \left(\frac{2w}{D} - 1\right)^{2p}$, [19], $f(w) = 1 - \left(\frac{w}{D} - stp(-i)\right)^{2p}$, [7], $f(w) = j(1 - ((w - 0.5)^2 + 0.75)p)$, [9],	$v(t) = \left\{ R_{\rm on} * \frac{w(t)}{D} + R_{\rm off} * \left(1 - \frac{w(t)}{D}\right) \right\} i(t)$
Simmons tunneling barrier [20]	$\frac{dw}{dt} = \begin{cases} f_{\rm off} \sinh\left(\frac{\lvert i \rvert}{i_{\rm off}}\right) e^{-e^{\frac{w - a_{\rm off}}{w_c} - \frac{\lvert i \rvert}{b}} - \frac{w}{w_c}}, & \text{if } i > 0 \\ -f_{\rm on} \sinh\left(\frac{\lvert i \rvert}{i_{\rm on}}\right) e^{-e^{\frac{a_{\rm on} - w}{w_c} - \frac{\lvert i \rvert}{b}} - \frac{w}{w_c}}, & \text{otherwise} \end{cases}$	Large system according [10], [20]–[22]
Yakopcic [8]	$\frac{dx}{dt} = g(v(t)) f(x(t))$, where $g(v) = \begin{cases} A_p (e^{v(t)} - e^{V_p}), & \text{if } v(t) > V_p \\ -A_n (e^{-v(t)} - e^{V_p}), & \text{if } v(t) < V_n \\ 0, & \text{otherwise} \end{cases}$ $f(x) = \begin{cases} e^{-\alpha_p (x - x_p)} \left(\frac{x_p - x}{1 - x_p} + 1\right), & \text{if } x > x_p \\ e^{\alpha_n (x + x_n - 1)} \left(\frac{x}{1 - x_n}\right), & \text{if } x < 1 - x_n \\ 1, & \text{otherwise} \end{cases}$	$i(t) = \begin{cases} a_1 x(t) \sinh(b v(t)), & \text{if } v(t) > 0 \\ a_2 x(t) \sinh(b v(t)), & \text{else} \end{cases}$
TEAM [12]	$\frac{dx}{dt} = \begin{cases} k_{\rm off} \left(\frac{i(t)}{i_{\rm off}} - 1\right)^{\alpha_{\rm off}} e^{-e^{\frac{x - a_{\rm off}}{w_c}}}, & \text{if } i > i_{\rm off} \\ k_{\rm on} \left(\frac{i(t)}{i_{\rm on}} - 1\right)^{\alpha_{\rm on}} e^{-e^{\frac{a_{\rm on} - x}{w_c}}}, & \text{if } i < i_{\rm on} \\ 0, & \text{otherwise} \end{cases}$	$\lambda = \log(R_{\rm off}/R_{\rm on})$ $v(t) = R_{\rm on} e^{\frac{\lambda}{w_{\rm off} - w_{\rm on}}(w - w_{\rm on})} i(t)$
Simplified Simmons barrier [12]	$\frac{dw}{dt} = \begin{cases} f_{\rm off} \sinh\left(\frac{\lvert i \rvert}{i_{\rm off}}\right) e^{-e^{\frac{w - a_{\rm off}}{w_c} - \frac{\lvert i \rvert}{b}} - \frac{w}{w_c}}, & \text{if } i > 0 \\ -f_{\rm on} \sinh\left(\frac{\lvert i \rvert}{i_{\rm on}}\right) e^{-e^{\frac{a_{\rm on} - w}{w_c} - \frac{\lvert i \rvert}{b}} - \frac{w}{w_c}}, & \text{else} \end{cases}$	$\lambda = \log(R_{\rm off}/R_{\rm on})$ $v(t) = R_{\rm on} e^{\frac{\lambda}{w_{\rm off} - w_{\rm on}}(w - w_{\rm on})} i(t)$
Eshraghian MIM with threshold [11]	$f(w) = s f_o + s f_m (1 - (2w - 1)^{2p})$ $\frac{dw}{dt} = \begin{cases} f_{\rm on} \left(1 - \frac{v}{2\phi_0}\right) e^{f(w)\phi_0 \left(1 - \sqrt{1 - \frac{v}{2\phi_0}}\right)}, & \text{if } v > 0, w < w_{\max} \\ -f_{\rm off} \left(1 + \frac{v}{2\phi_0}\right) e^{f(w)\phi_0 \left(1 - \sqrt{1 + \frac{v}{2\phi_0}}\right)}, & \text{if } v < 0, w > w_{\min} \\ 0, & \text{otherwise} \end{cases}$	$i(t) = a_1 w^n \sinh(b_1 v(t)) + a_2 (e^{b_2 v(t)} - 1)$

but also in Spectre. Furthermore, we have introduced systematic improvements in the circuit scheme of all the models that lead into important speedups in both SPICE and Spectre simulations, as detailed in Section V.

In order to explain the systematic improvements that we propose, let us first review the structure and the behavior of a basic memristor subcircuit (a netlist that is instantiated in a higher hierarchy circuit). Fig. 1 displays the three stages; a memristor subcircuit can be divided into.

First, the device *v–i* relationship is modeled as two terminals connected by a current generator in series with a resistor $R_{\rm aux}$. The current generator is governed by the expression

$$i = r(v, \vec{x}, \vec{a}) \tag{1}$$

where $\vec{x} = \{x_1, \ldots, x_q\}$ and $\vec{a} = \{a_1, \ldots, a_p\}$ are the state and auxiliary variables, respectively.

Second, for all models, the state variables evolution is determined by a set of differential equations, which depend on the voltage (voltage-controlled memristor) or the current (current-controlled memristor), the state and the auxiliary variables

$$\frac{dx_j}{dt} = f_j(v, i, \vec{x}, \vec{a}). \tag{2}$$

As seen in the central part of Fig. 1(a), with the aim of modeling each state variable, a current generator takes the value given by (2). A capacitor placed in series with the current generator performs the state variable integration. The circuit simulator internally performs the integration of the capacitor current. This way, the voltage at node x_j takes the form

$$x_j = \frac{1}{C} \int_{t_0}^{t} f_j(v, i, \vec{x}, \vec{a}) dt + v(x_j)_{|t_0}. \tag{3}$$

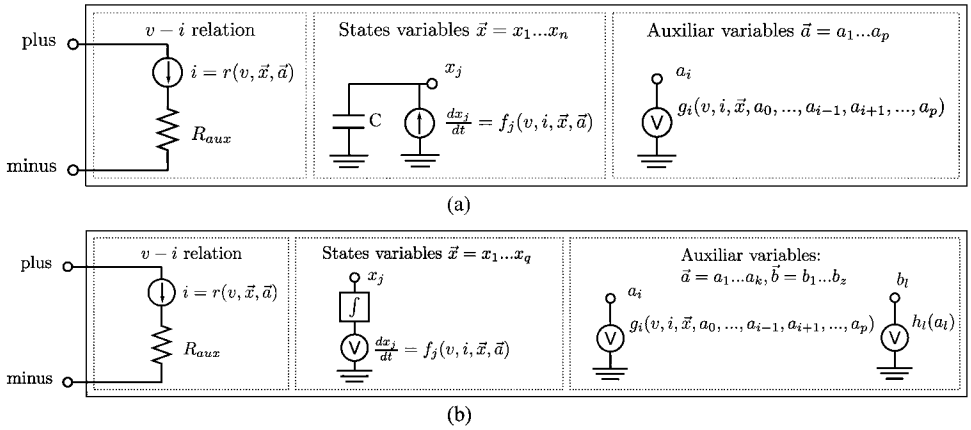

Fig. 1. Conventional [1(a)] memristor subcircuit scheme and the proposed scheme [1(b)]. (a) Previous subcircuit schemes. (b) MAF subcircuit scheme.

The x_j voltage is scaled by changing the capacitance C. Also, by specifying the initial voltage $v(x_j)_{|t_0}$ at the node x_j, the state variable initial condition gets fixed.

Finally, each of the variables that are not determined by a differential equation is modeled by a voltage source with the expression

$$a_i = g_i(v, i, \vec{x}, a_0, \ldots, a_{i-1}, a_{i+1}, \ldots, a_p) \tag{4}$$

Fig. 1(a) summarizes the previous scheme. Next, we describe the four systematic optimizations introduced in the subcircuits which are represented in Fig. 1(b).

1) Variable Integration Mechanism: The use of a capacitor together with a current generator to perform a variable integration makes the simulator manage two different components in each simulation step, which duplicates the convergence assessments.

SPICE-like simulators allow the utilization of several mathematical functions. Among these functions, the designer can choose integrators. By removing the capacitor–current generator set, and placing instead a single-voltage generator whose value refers to the differential equation, the circuit simulator only manages one component. Therefore, the computational load is reduced.

2) Variable Control: The proposed subcircuit schemes must provide the mechanism to ensure that some state variable values are constrained. Some subcircuit models [12] delegate in a set of diode-voltage generators to impose these bounds. Other subcircuits limit the ranges using switches [28]. This scheme involves several circuit components with the corresponding computational overhead.

In contrast, we can eliminate these components by using the simulator boundary functions. Furthermore, this mechanism can be integrated in the same definition as the differential equation modeling, simplifying the subcircuit structure. Consequently, we can reduce the computational load systematically eliminating some components. This way a state variable can be defined, initialized, and controlled in a single SPICE sentence:

$$\text{EW w 0 value} = \{\min(\text{min\_value}, \max(\text{idt}(\text{diff\_eq}, w\_init), max\_value))\}.$$

3) Common Values Aggregation: In case that during the computation of the state or auxiliary variables a value depending on an auxiliary variable is computed several times, the circuit simulator is performing the same operation multiple times. Let $h(a_l)$ be an operation evaluated several times. We propose to use a new variable b_l, which takes the form of a voltage source V_{b_l} in the subcircuit, instead of computing every time step the same value. This way, we have the following system:

$$\begin{aligned} v_m &= g_m(v, i, \vec{x}, a_0, \ldots, a_{m-1}, a_{m+1}, \ldots, a_p, h(a_l)) \\ &\vdots \\ v_n &= g_n(v, i, \vec{x}, a_0, \ldots, a_{n-1}, a_{n+1}, \ldots, a_p, h(a_l)) \end{aligned}$$

transformed into

$$\begin{aligned} b_l &= h_l(a_l) \\ v_m &= g_m(v, i, \vec{x}, a_0, \ldots, a_{m-1}, a_{m+1}, \ldots, a_p, b_l) \\ &\vdots \\ v_n &= g_n(v, i, \vec{x}, a_0, \ldots, a_{n-1}, a_{n+1}, \ldots, a_p, b_l), \end{aligned}$$

which creates the new set of variables $\vec{b} = \{b_0, \ldots, b_z\}$.

4) Current and Voltage Sources Aggregation and Disaggregation: In several cases, a state variable is modeled as a contribution of distinct voltage or current sources. If these sources do not have convergence problems, the circuit simulator needs to handle several components, which implies checking different convergence or boundary conditions. The aggregation into a single source can reduce the computational load. On the other side, there are sources computing complex functions that can lead into convergence problems. Those functions can be split to isolate the convergence problem.

IV. MAF

Continuing with the idea of aiding the design community, we have developed an MAF that encompasses all the stages necessary to include a memristor model in the standard circuit

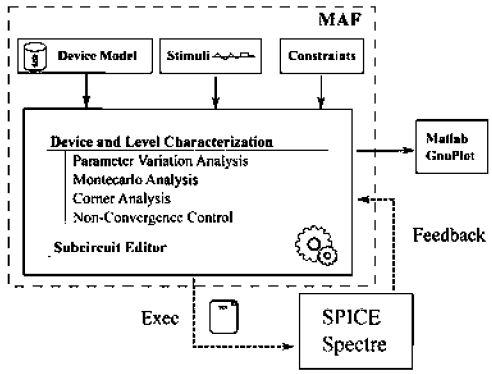

Fig. 2. MAF functional scheme, showing the main modules and the interaction with external software.

design flow. The MAF is developed in Java. Therefore, the application can be run in several OS.

As seen in Fig. 2, MAF functionalities are mainly covered by three main modules: *Device Characterization*, *Level Characterization*, and *Subcircuit Editor*. The first two are different simulators with which the user is able to fully characterize each of the memristor models included in the model library. Each module is illustrated with a simple example in Section IV-E.

The MAF model library includes the seven models described in Section II and their improved subcircuit netlists from Section III. It is likely that new memristor devices, and their corresponding models, will appear. The inclusion of these memristors in the models library is a key point in the extensibility of the framework. To deal with this problem, the MAF engine has been carefully designed as a modular system to allow the scalability of its functionalities. The MAF is composed of primitives which have been programmed to act as independent functional units, interacting between them by exchanging data models and scenarios.

Before describing MAF main modules, we will explain how both simulators—*Device Characterization* and *Level Characterization*—take advantage of MAF parameter variations handling capabilities. The framework helps analyzing the fluctuation of parameters which can destabilize the memristor behavior. For example, a slight variation in the threshold of the device may lead the memristor to act as a normal resistor. By contrast, under the presence of the spurious voltage, a value stored in a memristor can be altered. MAF simulators provide the way to study the dynamics of a specific memristor in the presence of parameter variations. We have included *Corner analysis* and *Monte Carlo* methods. In both methods, the variation of the voltage, temperature (depending on the memristor model), and model inherent parameters is given by a probabilistic deviation from the mean value. Therefore, instead of a specific memristor excited by a single voltage, several simulations are concurrently executed, performing a deeper analysis. This way, a wide set of scenarios which differ in the values of the device nominal parameters, the temperature (if present), and the voltage fluctuations are created and simulated.

A. Device Characterization Module

The *Device Characterization Module* is a transient simulator that follows the device behavior equations, providing a powerful environment to study the response of the models included in the database once stimulated by different patterns of the voltage and taking into account the parameter variations.

It is well known that some of these models require small time steps during the simulation in order to obtain valid results [10]. Additionally, the computation of some variables strongly depends on the timing of the simulation, so to avoid computation overflow problems and nondesired behaviors, our simulator guarantees that the time step is small enough to make the device behave correctly while ensuring the convergence of the simulation. The simulator also includes several techniques [29] to solve and recover from the nonconvergence problems derived from the variability of nanotechnologies, assuring the viability of the memristor behavior and the simulation convergence.

B. Level Characterization Module

The memristance presents a highly nonlinear relation with the flux. Therefore, the designer needs high accuracy in the flux value and related parameters. Thus, with the aim of setting a precise resistance value (e.g., to perform a multilevel storage or a weighted operation), the designer has to accurately know the amplitude and duration of the writing signal. With a fixed set of pulse amplitudes, the pulse length required to store a specific value will depend on the memristor speed.

For this reason, one of the main issues when designing memristive applications is, given a supply voltage, determining the time it takes for the device to perform an accurate state change. The *Level Characterization* module allows a comprehensive study of the memristance-flow relationship when writing a desired value in the memristor.

Consequently, this module helps in the study of the time evolution of memristance and model state variables. Also, the tool allows the automated calculation of the pulse lengths required to store the desired values.

Additionally, with this module, the designer is able to acquire and study the mean values and maximum deviations of the operation timing. This, together with the histogram and the state variable-memristance evolution plots, helps find the operation characteristics and define the safety margins.

C. Subcircuit Editor Module

This module loads the customized parameters brought from the characterized memristor and creates the appropriate subcircuit. In this stage, the user is able to modify the parameters to include additional components or to change the subcircuit structure. Furthermore, the designer can automatically generate the code to perform Monte Carlo & Corner analysis in the external circuit simulator. As a result, the generated subcircuit is ready to be referenced by the global circuit netlist.

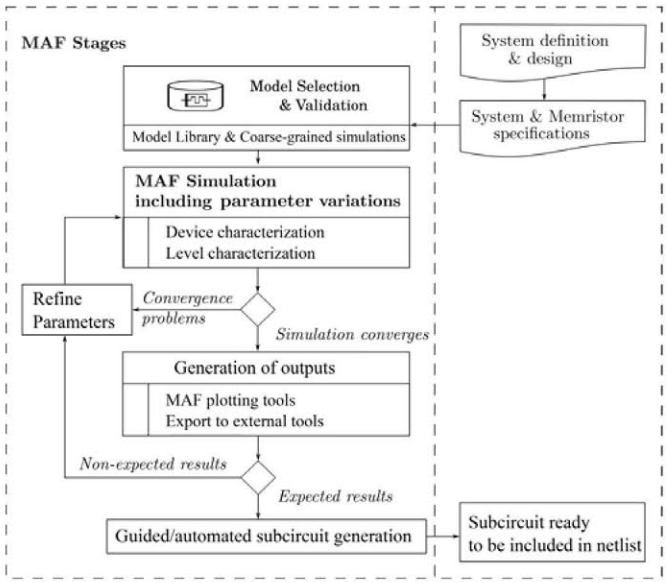

Fig. 3. Proposed methodology to be followed by the circuit designer.

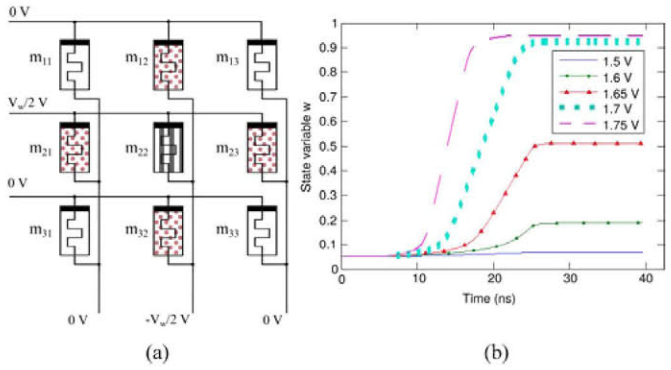

Fig. 4. (a) Crossbar circuit, showing the target cell (striped) and the alterable cells (dotted). (b) State variable w at different pulse volta ges. In all cases, the pulse lengths used are 20 ns with 8 ns slopes and a delay of 5 ns.

D. MAF Design Methodology

In this section, we put forward a recommended methodology, illustrated in Fig. 3, for the designers that wish to use the MAF.

The methodology departs from an application in which one or several memristors need to be used. The application has certain restrictions which translate into a set of constraints for the memristors. The designer is assumed to be acquaintance with the models so as to reduce the group of possible candidates. As seen in the scheme, the user can select the optimal model by comparing the behavior of the candidate models using coarse-grained MAF's *Device Characterization* simulations.

Once the memristor model and specifications are defined, in the next stage, the designer performs a refinement of the memristor's parameters, stimuli, timing, and powering conditions. This precise adjustment is performed employing MAF's *Device Characterization* and *Level Characterization* simulators. In case the analysis does not converge, the memristor is not fast enough, or if some parameter variation disrupts the operation, the parameters should be refined. The MAF offers its own graphical results representation, as well as formated-data exportation for the external software such as GnuPlot and MATLAB.

When the initial requirements are satisfied, a SPICE/Spectre subcircuit memristor netlist is generated, allowing the designer to customize it, include the subcircuit in the global circuit netlist, and proceed to its simulation.

E. Example of Use

To show the framework capabilities, we present a simple case of use targeting one of the most common memristive applications: a crossbar memory. More concretely, we will perform an eight-level writing operation in a 3×3 RRAM custom netlist, as shown in Fig. 4(a).

In this crossbar scheme, the writing operation involves two different pulse generators and nine cells, each composed of one memristor. Each memristor cell can be modified to store eight different resistance values, from 0 to 7, linearly separated. Initially, all cells have a value of "0." Rows and columns are numbered from 1 to 3. In this example, the desired operation will write a "7" in the second cell of the second row of the RRAM array (position $[2, 2]$). Crossbar array structures have structural problems. The main one refers to the spurious voltage which feeds the cells placed next to the selected RRAM. These voltages can modify the state variables in nondesired cells, and consequently, alter the stored values. To minimize those nondesired effects, we will use the half-voltage writing scheme shown in Fig. 4(a). Nonselected RRAM cells will be affected by lower spurious voltages.

Fig. 4(a) displays the circuit scheme; the target cell is filled with a striped pattern and cells that are sensitive to data alterations are filled with a dotted pattern.

The purpose of this example is twofold:

1) determining all the parameters to perform the full state change—including feeding voltage and timing characteristics.
2) analyzing the impact on the nonselected cells when the writing operation is performed.

Following the proposed methodology, first, we define the design constraints. In our example, we will assume that the circuit power supply limits the maximum voltage with which the memristor is fed, besides we set a limit in the duration of the writing operation; thus, we have the following.

1) Voltage supply should not exceed 2 V.
2) Operation should be performed in 15 ns.

The next step is selecting the most appropriate memristor model. We will use [11] with the default parameters because of its good tradeoff between accuracy and computational load, as well as the presence of a threshold in its behavior, which will help minimizing the cell data corruption effect.

We proceed to simulate in the MAF *Device Characterization* module different transient simulations to set the feeding voltage, taking into account that the model must be fast enough to perform the required operation. Fig. 4(b) shows simulation results. As can be seen, with 1.75 V, we can completely alter the state variable (bounded between "1" and "0"), and therefore,

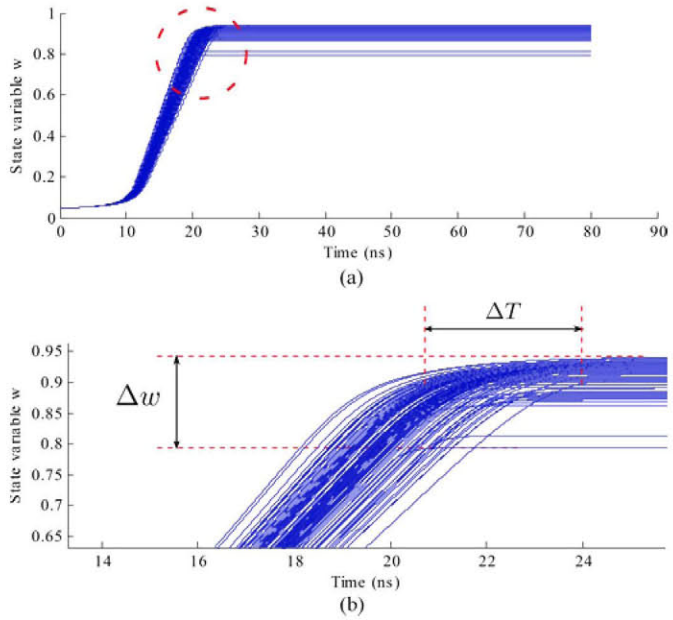

Fig. 5. State variable w in a Monte Carlo analysis, (a) general view and (b) detailed view showing ΔT and Δw margins due to variability. (a) General view of the state variable evolution. (b) Detail view of the state variable evolution.

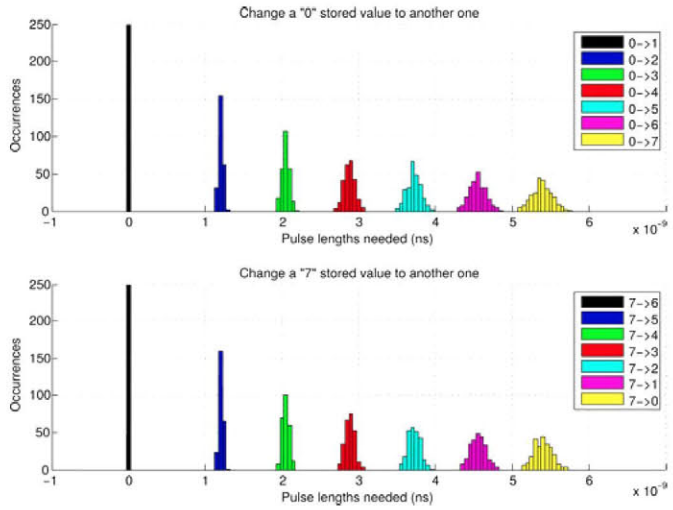

Fig. 6. Monte Carlo results given by the *MAF Level Characterization* module.

reach with this writing voltage either maximum/minimum memristance values.

Performing a Monte Carlo analysis with the *MAF Device Characterization* module, we study how the variation of the feeding voltage and other parameters affects the maximum storage levels (Δw) and the writing operation length (ΔT). Fig. 5 shows an example of the evolution of the state variable w along the time. In the detailed view, both ΔT and Δw are presented. Evaluating those values, we are able to refine the parameters of the memristor model and estimate the variability of the device.

Focusing on the pulse length of the writing operation, we carry out another Monte Carlo simulation using the *MAF Level Characterization* module. Fig. 6 shows the histogram with the pulse lengths occurrences required to perform the multilevel storage over 250 process scenarios. Based on this information, a designer is able to determine the operation security margins.

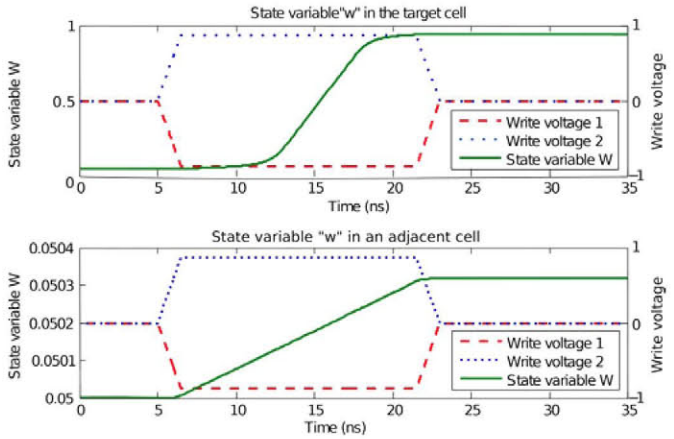

Fig. 7. SPICE simulation of the RRAM crossbar array. State variable evolution w in the target cell (the desired value is correctly stored) and in an adjacent cell (the state variable is almost unaltered).

After all the memristor variables have been settled, as well as the operation voltage and timing are chosen, using the *Subcircuit Editor* module, we automatically generate the corresponding SPICE subcircuit and proceed to include it in the global netlist. This subcircuit will be instantiated in each RRAM cell at the global SPICE netlist.

At this stage, we can verify in SPICE that the nondesired effects of a spurious alteration of previously stored data in adjacent cells do not appear. Fig. 7 displays the writing operation accomplished in both the target cell and an adjacent cell. As shown, the previous value stored in the adjacent cell is almost unaltered—notice the change in the left y-axis—thus, the impact is negligible.

V. Speed-Up Results

We have tested two different scenarios to illustrate the impact that the proposed subcircuit modifications have in the circuit simulator performance. The first scenario stands as the transient simulation of a sinusoidal voltage feeding a single memristor. The second scenario takes place as a 3×3 RRAM crossbar array. In this crossbar scheme, while the target is to write a single cell, the whole operation affects the nine memristors. In both cases, with the objective of increasing the simulation loads, the transient simulation step length has been limited to 1 ps. Dealing with the computational load of different models, it is impossible to guarantee a specific performance; however, with the computation of a large amount of steps, we extract a general trend that covers most cases and makes visible remarkable differences between models.

To broaden the scope of the analysis, we have performed the test in both *LTSpice* and *Spectre*.

The results show important speedups. In some cases, simulating the *Simmons tunneling barrier* memristor model, the

TABLE II
SPEED-UP RESULTS SUMMARY: SAVED TIME RATIO BETWEEN OUR PROPOSED SCHEME AND ORIGINAL MODELS

Model	Single Memristor LTSpice	Single Memristor Spectre	3x3 RRAM LTSpice
Non-linear	20.84%	3.15%	20%
Yakopcic	35.36%	0.01%	21.44%
Modified HP-Simmons	12.28%	22.83%	27.15%
HP-Simmons	NaN	NaN	45.87%
TEAM	30.01%	18.74%	29.33 %
Eshraghian	11.16%	3.57%	4.5%

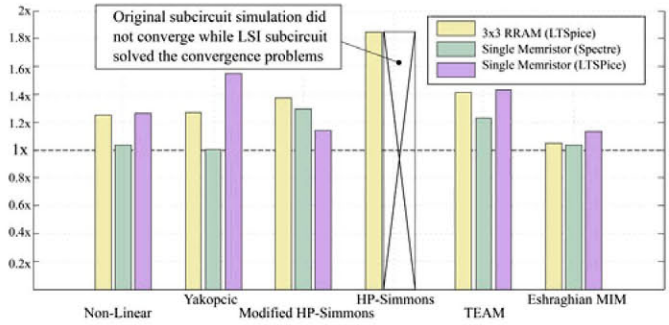

Fig. 8. Speed-up results. Note that model [10] did not converge before applying our proposed subcircuit modifications.

simulation could not converge using the conventional scheme subcircuit and time steps as short as 1 ps. On the contrary, when using our proposed subcircuit schemes, the simulation succeeded. In the cases where both conventional and new model simulations converge, we achieve a two-times speedup in a simulation using one or nine memristors. All simulations were run in an i7 950 (3.6 GHz) and 12-GB machine. Table II shows the percentage of time saved when using our proposed scheme instead of the conventional one, while Fig. 8 corresponds to the graphical representation of these speedups. As can be seen, the most complex models obtain a greater benefit of the proposed subcircuit modifications in the reduction of the computational load.

VI. CONCLUSION

Since the memristor was first built in 2008, a myriad of devices and models have appeared in the literature. However, the integration of these memristive devices in circuits is still not properly solved because their models are not mature and it is a hard task for the designer to control the convergence, accuracy, speed, and computational load of simulations. This study bridges the gap between technology scientists, memristor model developers, and circuit designers that want to incorporate memristors in their standard work flow. We have presented the MAF that is conceived to assist the memristor community throughout all design stages. It includes two simulators, i.e., *Device Characterization* and *Level Characterization*, that cover all the steps required to get a certain memristor model ready for application development, from model selection and validation to the fine adjustment of each parameter and variable even in the presence of process or conditions variation. The MAF automatically generates a SPICE/Spectre subcircuit of the analyzed memristor to ease its integration in application circuits.

The framework includes a library with the best-known memristor models and it is simple to introduce upcoming models. Additionally, systematic modifications have been carried out to improve the SPICE code of each model providing better convergence (solving scenarios where conventional subcircuits generate overflows) and significant simulations speedups. As illustrated by examples of use, the framework provides the designer with total simulation control, overcoming usual problems in the integration of memristors.

REFERENCES

[1] L. O. Chua, "Memristor—The missing circuit element," *IEEE Trans. Circuit Theory*, vol. CT-18, no. 5, pp. 507–519, Sep. 1971.

[2] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found." *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[3] C. Yakopcic, A. Sarangan, J. Gao, T. M. Taha, G. Subramanyam, and S. Rogers, "TiO 2 memristor devices," in *Proc. IEEE Nat. Aerosp. Electron. Conf.*, 2011, pp. 101–104.

[4] A. Mehonic, S. Cueff, M. Wojdak, S. Hudziak, O. Jambois, C. Labbe, B. Garrido, R. Rizk, and A. J. Kenyon, "Resistive switching in silicon suboxide films," *J. Appl. Phys.*, vol. 111, no. 7, pp. 074507-1–074507-9, 2012.

[5] J. P. Strachan, A. C. Torrezan, F. Miao, M. D. Pickett, J. J. Yang, W. Yi, G. Medeiros-Ribeiro, and R. S. Williams, "State dynamics and modeling of tantalum oxide memristors," *IEEE Trans. Electron. Devices*, vol. 60, no. 7, pp. 2194–2202, Jul. 2013.

[6] D. B. Strukov and R. S. Williams, "Exponential ionic drift: Fast switching and low volatility of thin-film memristors," *Appl. Phys. A*, vol. 94, no. 3, pp. 515–519, Nov. 2008.

[7] Z. Biolek, D. Biolek, and V. Biolková, "SPICE model of memristor with nonlinear dopant drift," *Radioengineering*, vol. 18, no. 2, pp. 210–214, 2009.

[8] C. Yakopcic, S. Member, T. M. Taha, G. Subramanyam, S. Member, R. E. Pino, and S. Rogers, "A memristor device model," *IEEE Electron Device Lett.*, vol. 32, no. 10, pp. 1436–1438, Oct. 2011.

[9] T. Prodromakis and B. P. Peh, "A versatile memristor model with nonlinear dopant kinetics," *IEEE Trans. Electron Devices*, vol. 58, no. 9, pp. 3099–3105, Sep. 2011.

[10] H. Abdalla and M. D. Pickett, "SPICE modeling of memristors," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2011, pp. 1832–1835.

[11] K. Eshraghian, O. Kavehei, K.-R. Cho, J. M. Chappell, A. Iqbal, S. F. Al-Sarawi, and D. Abbott, "Memristive device fundamentals and modeling: Applications to circuits and systems simulation," *Proc. IEEE*, vol. 100, no. 6, pp. 1991–2007, Jun. 2012.

[12] S. Kvatinsky, E. G. Friedman, A. Kolodny, S. Member, and U. C. Weiser, "TEAM: Threshold adaptive memristor model," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 1, pp. 211–221, Jan. 2013.

[13] D. Biolek, M. D. Ventra, Y. V. Pershin, and S. Member, "Reliable SPICE simulations of memristors, memcapacitors and meminductors," arXiv:1307.2717 [physics.comp-ph].

[14] I. Ebong and P. Mazumder, "Self-controlled writing and erasing in a memristor crossbar memory," *IEEE Trans. Nanotechnol.*, vol. 10, no. 6, pp. 1454–1463, Nov. 2011.

[15] S. Ham, H. Mo, and K. Min, "Low-power VDD/3 write scheme with inversion coding circuit for complementary memristor array," *IEEE Trans. Nanotechnol.*, vol. 12, no. 5, pp. 851–857, Sep. 2013.

[16] C. Yakopcic, T. M. Taha, G. Subramanyam, and S. Rogers, "Multiple memristor read and write circuit for neuromorphic applications," in *Proc. Int. Joint Conf. Neural Netw.*, 2011, pp. 2676–2682.

[17] J. Cong and X. Bingjun, "mrFPGA: A novel FPGA architecture with memristor-based reconfiguration," in *Proc Symp.—Quart. J. Mod. Foreign Literatures*, Jun. 2011, pp. 1, 8–9.

[18] L. Gao, F. Alibart, and D. Strukov, "Programmable CMOS/memristor threshold logic," *IEEE Trans. Nanotechnol.*, vol. 12, no. 2, pp. 115–119, Mar. 2012.

[19] Y. N. Joglekar and S. J. Wolf, "The elusive memristor: Properties of basic electrical circuits," *Eur. J. Phys.*, vol. 30, no. 4, pp. 661–683, 2009.
[20] M. D. Pickett, D. B. Strukov, J. L. Borghetti, J. J. Yang, G. S. Snider, D. R. Stewart, and R. S. Williams, "Switching dynamics in titanium dioxide memristive devices," *J. Appl. Phys.*, vol. 106, no. 7, pp. 074508-1–074508-6, 2009.
[21] J. G. Simmons, "Generalized formula for the electric tunnel effect between similar electrodes separated by a thin insulating film," *J. Appl. Phys.*, vol. 34, no. 6, pp. 1793–1803, 1963.
[22] J. Simmons, "Electric tunnel effect between dissimilar electrodes separated by a thin insulating film," *J. Appl. Phys.*, vol. 34, pp. 2581–2590, 1963.
[23] D. Biolek, Z. Biolek, and V. Biolková, "SPICE modeling of memristive, memcapacitative and meminductive systems," in *Proc. Eur. Conf. Circuit Theory Des.*, Aug. 2009, pp. 249–252.
[24] G. Snider, "Cortical computing with memristive nanodevices," *SciDAC Rev.*, vol. 10, pp. 58–65, 2008.
[25] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnol.*, vol. 3, no. 7, pp. 429–33, Jul. 2008.
[26] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano lett.*, vol. 10, no. 4, pp. 1297–301, Apr. 2010.
[27] A. S. Oblea, A. Timilsina, D. Moore, K. A. Campbell, and S. Member, "Silver chalcogenide based memristor devices," in *Proc. Int. Joint Conf. Neural Netw.*, 2010, vol. 3, pp. 4–6.
[28] A. Rak and G. Cserey, "Macromodeling of the memristor in SPICE," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 29, no. 4, pp. 632–636, Apr. 2010.
[29] F. Garcia-Redondo, M. López-Vallejo, P. Ituero, and C. L. Barrio, "A CAD framework for the characterization and use of memristor models," in *Proc. IEEE Int. Conf. Synth., Model., Anal. Simul. Methods Appl. Circuit Des.*, 2012, pp. 25–28.

Fernando García-Redondo received the Graduate degree from the Technical University of Madrid, Madrid, Spain, in 2011, with a degree (B.S. + M.Sc.) in telecommunication engineering (electrical engineering). He received the Master of Science degree in electrical engineering (major in electronics) from the Technical University of Madrid in 2012, where he is currently working toward the Ph.D. degree at the Integrated Systems Laboratory.

His main research interests include the hard radiation design and memristor modeling.

Marisa López-Vallejo (M'00) received the M.S. and Ph.D. degrees from the Universidad Politécnica de Madrid, Madrid, Spain, in 1993 and 1999, respectively.

She is an Associate Professor with the Department of Electronic Engineering, Universidad Politécnica de Madrid. She was with the Lucent Technologies, Bell Laboratories, Murray Hill, NJ, USA, as a Technical Staff Member. Her current research interests include low-power, process voltage, temperature-aware designs, computer-aided diagnostic methods and tools, and application-specific high-performance programmable architectures.

Pablo Ituero received the 5-year engineering degree (B.S. + M.Sc.) and the Ph.D. degree in telecommunications with a major in electronics from the Universidad Politécnica de Madrid, Madrid, Spain, and the M.Sc. degree in electrical engineering with a major in system-on-a-chip-design from the Royal Institute of Technology of Sweden, Stockholm, Sweden.

He is currently an Assistant Professor (Prof. Ayudante Doctor) with the Department of Electronic Engineering, Universidad Politécnica de Madrid. During 2005–2008, he was with an FPU scholarship of the Spanish Ministry of Education and Science.